

# *Spring with WSDLS/java2wsdl*

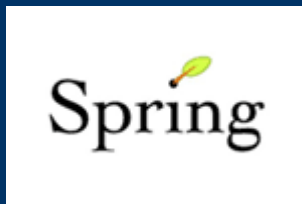
“He is like superman against logic bullets!”

-A grumpy guy in the hallway in reference to one of the stakeholders

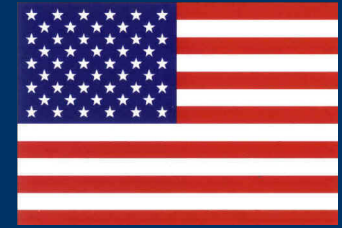


# *What is Spring and why use it?*

- Lightweight container framework
- Inversion of Control
- Aspect-orientated
- Manages life cycle



# Why do I care? The good!



- Good software engineering practices for modern service orientated architectures.
- Easy to integrate web-services into framework.
  - JMS/RMI/HTTP
  - Integrates easily with middleware such as web-methods
  - Easy integration with WSDL's
- (You don't)



# Example WSDL (taken from

<http://mirror.facebook.com/mozdev.org/spengler/BabelFishService.wsdl.xml>)

```
<?xml version="1.0" ?>
- <definitions name="BabelFishService" xmlns:tns="http://www.xmethods.net/sd/BabelFishService.wsdl"
  targetNamespace="http://www.xmethods.net/sd/BabelFishService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
- <message name="BabelFishRequest">
  <part name="translationmode" type="xsd:string" />
  <part name="sourcedata" type="xsd:string" />
</message>
- <message name="BabelFishResponse">
  <part name="return" type="xsd:string" />
</message>
- <portType name="BabelFishPortType">
- <operation name="BabelFish">
  <input message="tns:BabelFishRequest" />
  <output message="tns:BabelFishResponse" />
</operation>
</portType>
- <binding name="BabelFishBinding" type="tns:BabelFishPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="BabelFish">
  <soap:operation soapAction="urn:xmethodsBabelFish#BabelFish" />
- <input>
  <soap:body use="encoded" namespace="urn:xmethodsBabelFish"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</input>
- <output>
  <soap:body use="encoded" namespace="urn:xmethodsBabelFish"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</output>
</operation>
</binding>
- <service name="BabelFishService">
  <documentation>Translates text of up to 5k in length, between a variety of languages.</documentation>
- <port name="BabelFishPort" binding="tns:BabelFishBinding">
  <soap:address location="http://services.xmethods.net:80/perl/soaplite.cgi" />
</port>
</service>
</definitions>
```

Inputs

Outputs

Identify All Operations

# *Create an Interface for the WSDL*

- Forces good design practices

```
import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface BabelFishIF extends Remote {  
    public String BabelFish(String translationMode, String sourceData) throws RemoteException;  
}
```

# Setting up the service in Spring

```
<bean id="babelFish"
  class="org.springframework.remoting.jaxrpc.JaxRpcPortProxyFactoryBean">
  <property name="serviceInterface">
    <value>exp.client.BabelFishIF</value>
  </property>
  <property name="portInterface">
    <value>exp.client.BabelFishIF</value>
  </property>
  <property name="wsdlDocumentUrl">
    <value>http://www.xmethods.com/sd/2001/BabelFishService.wsdl</value>
  </property>
  <property name="namespaceUri">
    <value>http://mirror.facebook.com/mozdev.org/spengler/BabelFishService.wsdl.xml</value>
  </property>
  <property name="serviceName">
    <value>BabelFishService</value>
  </property>
  <property name="portName">
    <value>BabelFishPort</value>
  </property>
  <property name="serviceFactoryClass">
    <value>org.apache.axis.client.ServiceFactory</value>
  </property>
</bean>
```

# Using it!

- So easy, an HR Manager could do use it! (doubtful)

```
ApplicationContext context = new FileSystemXmlApplicationContext("path$xmlwemade.xml")
BabelFishIF babelFish = (BabelFishService) context.getBean(babelFish);
String translateThis = babelFish.BabelFish("en_fr", "Spring is very cool!");
```

We get: Le ressort est très frais !

Or to dutch: De lente is zeer koel!

# Using WSDL's without Spring

```
String wsdlDocumentUrl = "URL2WSDL.wsdl";
String namespaceUri = "http://www.coulditbe.com/path/to/namespace.wsdl";
String serviceName = "BabelFishService";
String portName = "BabelFishPort";
QName serviceQN = new QName(namespaceUri, serviceName);
QName portQN = new QName(namespaceUri, portName);

ServiceFactory serviceFactory = ServiceFactory.newInstance();

Service service = serviceFactory.createService(new URL(wsdlDocumentUrl) , service QN);
BabelFishIF babelFish = (BabelFishRemote) service.getPort(BabelFishIF.class, portQN);
```

# *The propaganda, the bad!*



- Speed
  - We could be doing binary transmissions.
    - But no one cares because application runtime is relatively large.
- Extra time configuring spring.
- Others you guys think?



# Quick overview of Java2Wsd1

- Define an interface (like BabelFishIF)
- Create the WSDL with Java2Wsd1

```
java org.apache.axis.wsdl.Java2WSDL -o outputwsdlfile.wsdl  
  -l "location of the service"  
  -n "target namespace"
```

- <http://ws.apache.org/axis/java/reference.html#Java>
- 
- Create bindings:

```
java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -S true wsdlname.wsdl
```